Postel

RFC:767

A STRUCTURED FORMAT FOR TRANSMISSION OF MULTI-MEDIA DOCUMENTS

Jonathan B. Postel

August 1980

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

(213) 822-1511

< INC-PROJECT, MMMSFS.NLS.21, >, 5-Sep-80 20:19 JBP ;;;;

Postel

TABLE OF CONTENTS

PREFACE iii

1. INTRODUCTION 1

1.1. Motivation 1
1.2. Scope 1
1.3. Terminology 1
1.4. Document Description 2
1.5. Other Work 2

2. SPECIFICATION 3

2.1. Document 3
2.2. Message Objects 5
2.3. Body Structures 13
2.3.1. Simple Elements 13
2.3.2. Structured Text 13
2.3.3. NLS File Example 13
2.3.4. Multimedia Structures 15
2.3.5. The Media 21
2.3.6. TEXT 22
2.3.7. VOICE 22
2.3.8. FACSIMILE 23
2.3.9. GRAPHICS 24

3. EXAMPLES & SCENARIOS 25

Example 1: Text Example 25
Example 2: Multimedia Example 28

REFERENCES 31

PREFACE

This is the first edition of this format specification and should be treated as a request for comments, advice, and suggestions. A great deal of prior work has been done on computer aided message systems and some of this is listed in the reference section. This specification was shaped by many discussions with members of the ARPA research community, and others interested in the development of computer aided message systems. This document was prepared as part of the ARPA sponsored Internetwork Concepts Research Project at ISI.

Jon Postel

J. Postel
USC-ISI
August 1980

A STRUCTURED FORMAT FOR TRANSMISSION OF MULTI-MEDIA DOCUMENTS

1. INTRODUCTION

This document describes a format for transmitting structured data representations of multimedia documents. This format is intended to be used with the Internet Message Protocol in an internetwork message delivery system. That system is designed to transmit messages between processes in host computers called Message Processing Modules (MPMs). MPMs are located in several networks and together constitute an internetwork message delivery system. The Internet Message Protocol defines a message as being composed of an Identification, a Command, and a Document. This report is intended to define the format of such Documents. The reader is assumed to be familiar with the Internet Message Protocol [1].

1.1. Motivation

Computer applications are being implemented which interact with users in a variety of media (text, graphics, facsimile, speech). As computer devices become available to process multimedia information it becomes desirable to use computers to exchange multimedia information between programs and users via various mechanisms including computer mail.

1.2. Scope

This format is intended to be used for the transmission of multimedia documents in the internetwork message delivery system, but it is thought that it has a wider applicability.

1.3. Terminology

The messages are routed by a process called the Message Processing Module or MPM. Messages are created and consumed by User Interface Programs (UIPs) in conjunction with users.

The basic unit transferred between MPMs is called a message. A message is made up of a transaction identifier (which uniquely identifies the message), a command (which contains the necessary information for delivery), and document. The document is a data structure.

For a personal letter the document body corresponds to the contents of

the letter; the document header corresponds to the date line, greeting, and signature.

For an inter-office memo the document body corresponds to the text; the document header corresponds to the header of the memo.

The commands correspond to the information used by the Post Office or the mail room to route the letter or memo. Some of the information in the command is supplied by the UIP.

1.4. Document Description

The document is composed of fields. Each field will carry an identifying name. Typical fields are DATE, TO, SUBJECT, and BODY. Most of the fields will be very simple, some will be complex. The body field may be quite complex. For example, the DATE is a very constrained character string specifying the date and time in ISO format. A more complex example is the TO field which is a list of mailboxes, where a mailbox is itself a property list of address information items. The BODY may be simply a character string, or a very structured collection of data representing information in different media.

The BODY may be structured to indicate a controlled presentation of multimedia information. There is provision for the inclusion of text, graphics, facsimile, and voice information in the body of documents. The presentation of information units may sequential, independent, or simultaneous.

1.5. Other Work

This protocol the benefited from the earlier work on message protocols in the ARPA Network [2,3,4,5,6], and the ideas of others about the design of computer message systems [7,8,9,10,11,12,13,14,15,16,17,18].

2. SPECIFICATION

The structured format of a document is built on the basic data elements used in the Internet Message Protocol [1].

2.1. Document

The document is a property list of <name,value> pairs called fields. A few fields are specifically required and many are optional. Some of the field values are simple and a few are quite complicated. In particular the body value may be highly structured.

Older message systems have considered the document to be divided into a header and a body, and have used keywords to indicate specific header fields (e.g., date, to, subject). Roughly speaking, this functionality is provided in this new structured format by considering the name part of the <name,value> pair to be a keyword. In addition, this new structured format eliminates the separate treatment of the body.

It is impossible to foresee the many forms documents will take so the standard for a document header must be flexible. The approach here is to define a set of basic fields and allow addition of whatever fields are necessary. Features added in this fashion may not be understood by others.

The minimum document is a property list of the following fields:

Name	Value
DATE	date string (name)
SENDER	a mailbox
SUBJECT	subject string (text)
BODY	a data structure

A typical document is a property list containing the following fields:

Name	Value
DATE	date string (name)
SENDER	a mailbox
FROM	list of mailboxes
TO	list of mailboxes
CC	list of mailboxes
SUBJECT	subject string (text)
BODY	a data structure

An elaborate document might contain the following fields:

Name	Value
DATE	date string (name)
SENDER	a mailbox
FROM	list of mailboxes
TO	list of mailboxes
CC	list of mailboxes
BCC	list of mailboxes
REPLY-TO	list of mailboxes
SUBJECT	subject string (text)
COMMENTS	comment string (text)
MESSAGE-ID	message identifier of this message (text)
IN-REPLY-TO	message identifier of previous message (text)
REFERENCES	message identifiers of other messages (text)
KEYWORDS	key terms used in this message (text)
BODY	a data structure

One of the key objects is the mailbox. It appears in the sender, from, to, cc, bcc, and reply-to fields. The mailbox is a property list of objects that combine to specify a destination recipient for a message. Most of the <name,value> pairs that make up a mailbox are identical to those used in the deliver command in the Internet Message Protocol [1]. A few additional <name,value> pairs are defined for use in a mailbox in the document context. In particular, there is a field for the real name of a person in contrast to the "user name" which identifies a computer account.

In addition there is a field to specify a distribution group name. Such group names are used to indicate that a document is being sent to a group of recipients. This essentially presents an alternate form for a mailbox which consists of the single <name,value> pair for the group name. There is no required relationship between a group name mailbox and other mailboxes in the same list.

For example, all of the following situations are allowed:

- a mailbox list consisting of a single mailbox specifying a particular user,

- a mailbox list consisting of a single mailbox with a group name,

- a mailbox list consisting of a mailbox with a group name and a mailbox specifying a particular user, with either the user in or not in the group,

- a mailbox list consisting of a mailbox with a group name and a

several mailboxes specifying a particular users, with some users in the group and some not,

- a mailbox list consisting of several mailboxes specifying group names and a several mailboxes specifying a particular users, with some users in the groups and some not.

2.2. Message Objects

In the documents of messages, we use a set of objects such as mailbox or date. These objects are encoded in basic data elements. Some objects are simple things like integers or character strings, other objects are more complex things like lists or property lists. The following is a list of the objects used in messages. The object descriptions are in alphabetical order.

Account

The account information. Represented by a name element.

Address

Address is intended to contain the minimum information necessary to identify a user, and no more (compare with mailbox).

An address is a property list which contains the following <name,value> pairs:

name	description
NET	network name
HOST	host name
USER	user name

or:

name	description
MPM	mpm-identifier
USER	user name

Answer

A yes (true) or no (false) answer to a question. Represented by a boolean element.

BCC

A list of mailboxes. The addresses of those who receive "blind carbon copies" of the message.

Body

A data structure. This may be as simple as a character string (represented by a name or text element), or complex structure of lists. It may be encrypted in part or in whole. Section 3.3 describes some possible structured bodies.

C

A character. Represented by a name element.

CC

A list of mailboxes. When copies of a message are sent to others in addition to the addresses in the To object, those to whom the copies are sent will have their addresses recorded here.

City

A city. Represented by a name element.

Comments

A comment string. Represented by a text element.

Count

A count of items of some sort. Represented by a integer element.

Country

A country. Represented by a name element.

Date

The date and time are represented according to the International Standards Organization (ISO) recommendations [19,20,21]. Taken together the ISO recommendations 2014, 3307, and 4031 result in the following representation of the date and time:

yyyy-mm-dd-hh:mm:ss,fff+hh:mm

Where yyyy is the four-digit year, mm is the two-digit month, dd is the two-digit day, hh is the two-digit hour in 24 hour time, mm is the two-digit minute, ss is the two-digit second, and fff is the decimal fraction of the second. To this basic date and time is appended the offset from Greenwich as plus or minus hh hours and mm minutes.

The time is local time and the offset is the difference between local time and Coordinated Universal Time (UTC). To convert from local time to UTC algebraically subtract the offset from the local time.

For example, when the time in
Los Angeles is 14:25:00-08:00
the UTC is 22:25:00

or when the time in
Paris is 11:43:00+01:00
the UTC is 10:43:00

Device

A device name. Represented by a name element.

ment

roperty list of fields.

Distribution Group

An distribution group is a property list which contains the following <name,value> pair:

name	description
GROUP	document distribution group name

This construct is used so that a distribution group will be a special case of a mailbox.

Facsimile Structure

A facsimile data structure. Represented by a property list.

File

A file name. Represented by a name element.

Format

A format indicator. Represented by a name element.

From

A list of mailboxes. The From is the name of the author of a document.

Graphics Structure

A graphics data structure. Represented by a property list.

Group

A document distribution group name. Represented by a name element.

Host

A host name. Represented by a name element.

Ident

The identifier of a person, usually their initials. Represented by a name element.

In-Reply-To

The message identifier of previous message. Represented by a text element.

Internet Address

This identifies a host in the ARPA internetwork environment. The internet address is a 32 bit number, the higher order 8 bits identify the network, and the lower order 24 bits identify the host on that network [22]. For use in this format the internet address is divided into eight bit fields and the value of each field is represented in decimal digits. For example, the ARPANET address of ISIE is 167837748 and is represented as 10,1,0,52. Further, this

representation may be extended to include an address within a host, such as the TCP port of an MPM, for example, 10,1,0,52,0,45.

Keywords

The key terms used in this message. Represented by a text element.

Mailbox

This is the destination address of a user of the internetwork mail system. Mailbox contains information such as network, host, location, and local user identifier of the recipient of the message. The mailbox may contain information in addition to the minimum required for delivery.

As an example, when one sends a message to someone for the first time, he may include many items to aid in identifying the correct recipient. However, once he gets a reply to this message, the reply will contain an Address (as opposed to Mailbox) which may be used from then on.

A mailbox is a property list. A mailbox might contain the following <name,value> pairs:

name	description
MPM	mpm-identifier
NET	network name
HOST	host name
PORT	address of MPM within the host
USER	user name (computer account name)
PERSON	the real name of a person
GROUP	document distribution group
ORG	organization name
CITY	city
STATE	state
COUNTRY	country
ZIP	zip code
PHONE	phone number

The minimum mail box is an Address or a Distribution Group.

Message-ID

The message identifier of this message. This is not related to the MPM message identification, but is a UIP long term document identifier. Represented by a text element.

MPM-Identifier

The internetwork address of an MPM. This may be the ARPA Internet Address or an X.121 Public Data Network Address [23]. The mpm-identifier is a property list which has one <name,value> pair. This unusual structure is used so that it will be easy to determine the type of address used.

Net

A network name. Represented by a name element.

NLS Block

The information in an NLS node. Represented by a property list.

NLS Node

An NLS block and substructure. Represented by a property list.

NLS Substructure

A list of NLS nodes. Represented by a list.

Org

An organization name. Represented by a name element.

Paragraph

A paragraph of text. Represented by a text element.

Parcel

The basic unit of voice data. Represented by a bitstr element.

Person

The real name of a person. Represented by a name element.

Password

A password. Represented by a name element.

Phone

A phone number. Represented by a name element.

Pointer

A pointer to information stored outside this data structure. A property list containing the information necessary to locate the external data, the information necessary to gain access to the external data, and the information necessary to apply the correct interpretation to the external data. For example, this might include:

name	description
NET	network name
HOST	host name
FILE	file name
USER	user name (computer account name)
PASSWORD	password
ACCOUNT	account
FORMAT	format

Port

The address of MPM within the host. Represented by a name element.

Presentation Descriptor

A property list of <name,value> pairs, where the name is an order indicator, and the value is a presentation element. The order indicators are SEQUENTIAL, SIMULTANEOUS, and INDEPENDENT.

Presentation Element

A property list of media structures.

Protocol

The name of the coding scheme used for a medium. Represented by a name element.

References

The message identifiers of other messages. Represented by a list of text elements.

Reply-To

A list of mailboxes. Sometimes it will be desired to direct the replies of a message to some address other than the from or the sender. In such a case the reply-to object can be used.

R 450 Block

The unit of Rapicom 450 data (585 bits). Represented by a bitstr element.

Sender

A mailbox. The sender will contain the address of the individual who sent the message. In some cases this is NOT the same as the author of the message. Under such a condition, the author should be specified in the from object.

SID

An NLS statement indetifier. Represented by a integer element.

State

A state name. Represented by a name element.

Subject

The subject of the message. Represented by a text element.

Text Structure

A text data structure. Represented by a property list.

To

A list of mailboxes. To identifies the addressees of the message.

User

A user name (computer account name). Represented by a name element.

Version

A version number. Represented by a index element.

Vocoder

A vocoder name. Represented by a name element.

Voice Structure

A voice data structure. Represented by a property list.

X121 Address

This identifies a host in the Public Data Network environment. When used as a part of identifier, it identifies the originating host of a message. The X121 address is a sequence of up to 14 digits [23]. For use in this format the X121 address is represented in decimal digits.

ZIP

A zip code. Represented by a name element.

2.3. Body Structures

2.3.1. Simple Elements

The body could simply be a single data element. For example a single text element can represent a lengthy character string.

<body> := TEXT

or

text:"this is the actual text of the body"

2.3.2. Structured Text

The body could be thought of as paragraphs, where each paragraph is represented by a text element. The paragraphs are then the elements of a list.

<body> := LIST (<paragraph>, <paragraph>, ...)

<paragraph> := TEXT

or

list:(text:"paragraph one", text:"paragraph two", ...)

2.3.3. NLS File Example

It is possible to represent the data from NLS files in this format. NLS is a large multipurpose system which operates on structured data files. The files are tree structured, and there is data associated with each node of the tree. There are several fields associated with each node as well.

An NLS file is:

```
proplist(                                                    file
  name:"FILENAME", name:<file>                          name of file
  name:"CREATION-DATE", name:<date>               creation date and time
  name:"VERSION", index:<version>                    file version number
  name:"SID-COUNT", integer<count>                     current SID count
  name:"LAST-WRITER", name:<ident>                   last writer of file
  name:"OWNER", name:<ident>                               owner of file
  name:"LAST-WRITE-TIME", name:<date>          last write date and time
  name:"LEFT-NAME-DELIM-DEFAULT", name:<c>                 default name
  name:"RIGHT-NAME-DELIM-DEFAULT", name:<c>                  delimiters
  name:"SUBSTRUCTURE", <nls-substructure>                   substructure
)endlist
```

An NLS substructure is:

```
list:(                                                     substructure
  <nls-node>                                        node is defined below
    .
    .
    .
)endlist
```

An NLS node is:

```
proplist:(                                                         node
  name:"BLOCK", <nls-block>                          block defined below
  name:"SUBSTRUCTURE", <nls-substructure>                 substructure
)endlist
```

An NLS block is:

```
proplist:(                                                        block
  name:"LEFT-NAME-DELIM", name:<c>                   left name delimiter
  name:"RIGHT-NAME-DELIM", name:<c>                 right name delimiter
  name:"SID", integer:<sid>                                  SID number
  name:"CREATOR", name:<ident>                     statement creator
  name:"CREATION-TIME", name:<date>              creation date and time
  name:"DATA", <data>                               data defined below
)endlist
```

NLS data is:

```
proplist:(                                                    data
  name:"<a data name>", <type depends on data name>
                .                .
                .                .
                .                .
)endlist
```

For text, data is:

```
proplist:(                                                    data
  name:"TEXT", text:"text of statement"                       text
)endlist
```

2.3.4. Multimedia Structures

One can conceive of graphical information being displayed along with a running commentary, much as seminars use slides. A slide and its description are tied together. The coordination of such a presentation is central to its understanding. This synchronization should be captured within the document structure.

There are three fundamentally different types of time ordered control which are needed within the document structure. These are:

Simultaneous
Sequential
Independent

Simultaneous data is intended for synchronous presentation. The implication is that this data is presented in parallel.

Sequential data items will be presented one at a time, in the order listed. The ordering is strictly left to right.

Independent data can be presented in any time order. It is not ordered in any manner.

The data is broken into small information units called presentation elements or PEs. The PEs can be combined in structures to control the presentation order. A PE is a property list of elements representing information of various media. For example:

```
<pe> := proplist(
          name:"VOICE", <voice-structure>,
          name:"GRAPHICS", <graphics-structure>
        )endlist
```

PEs are combined into larger controled presentations by presentation-descriptors or PDs. A PD is a property list which specifies the type of time ordering of the PEs in its list.

```
<pd> := <<seq>> | <<sim>> | <<ind>>

<<seq>> := name:"SEQUENTIAL", <pe>

<<sim>> := name:"SIMULTANEOUS", <pe>

<<ind>> := name:"INDEPENDENT", <pe>
```

A PE is a property list of the media <name,value> pairs, or PDs.

```
<pe> := <<text>> | <<voice>> | <<facsimile>>
      | <<graphics>> | <pd>

<<text>> := name:"TEXT", <text structure>

<<voice>> := name:"VOICE", <voice structure>

<<facsimile>> := name:"FACSIMILE", <facsimile structure>

<<graphics>> := name:"GRAPHICS", <graphics structure>
```

If more than one <name,value> pair is present within a PE the media are presented on different output devices in the order specified by the PE's parent PD. The order of appearance within the proplist is important only in the event that the parent PD specified sequential ordering.

The structure of multimedia messages which use this scheme will be demonstrated by a few simple examples chosen to illustrate a basic text document and the different ordering options. The last example will suggest some more exotic uses.

Plain Text Message

A simple text body could be represented in a single text data structure. To give the simplest example of a structured body we show a simple text body represented in the multimedia structure.

```
<body> := <pd>

  <pd> := <<seq>>

    <<seq>> :=  name:"SEQUENTIAL", <pe>

      <pe> := name:"TEXT", <text structure>

or

proplist: (name:"SEQUENTIAL",
           proplist:(
             name:"TEXT", <text structure>
           )endlist
)endlist
```

Simultaneous Ordering

This ordering option is used to indicate when separate streams are to be presented in parallel. For example, assume GRAPHICS and VOICE data were to be presented using simultaneously.

```
<body> := <pd>

  <pd> := <<sim>>

    <<sim>> :=  name:"SIMULTANEOUS", <pe>

      <pe> := name:"VOICE", <voice structure>
              name:"GRAPHICS", <graphics structure>

or

proplist:(
  name:"SIMULTANEOUS",
    proplist:(
      name:"VOICE", <voice structure>
      name:"GRAPHICS", <graphics structure>
    )endlist
)endlist
```

Sequential Ordering

This option is used to indicate sequential time ordering. The media in the sub-tree below this PD are not separate streams. Using again the example above, assume GRAPHICS and VOICE data were to be presented using sequential ordering.

```
<body> := <pd>

  <pd> := <<seq>>

    <<seq>> :=  name:"SEQUENTIAL", <pe>

      <pe> := name:"VOICE", <voice structure>
              name:"GRAPHICS", <graphics structure>
```

or

```
proplist:(
  name:"SEQUENTIAL",
    proplist:(
      name:"VOICE", <voice structure>
      name:"GRAPHICS", <graphics structure>
    )endlist
  )endlist
```

Independent Ordering

It is apparent that some output devices are very slow in comparison to others. An example which demonstrates this is facsimile. The majority of facsimile devices are slow. A detailed picture transmitted at 9600 baud takes minutes to print. It is inconvenient for the user to wait on such a device when the voice or text information which accompanies it is short.

For example, if the document a facsimile image and the text "Hello Frank, here's a copy of that picture you requested." The user need not wait for the picture. The facsimile machine might be spooled, in which case he would pick up the picture later. In a sense the picture was time independent of the text.

```
<body> := <pd>

  <pd> := <<ind>>

    <<ind>> :=  name:"INDEPENDENT", <pe>

      <pe> := name:"FACSIMILE", <facsimile structure>
              name:"TEXT", <text structure>

or

proplist:(
  name:"INDEPENDENT",
    proplist:(
      name:"FACSIMILE", <facsimile structure>
      name:"TEXT", <text structure>
    )endlist
)endlist
```

A Stream Example

By making use of the structure and the sequential ordering option it is possible to initiate a stream. The stream will proceed at its own pace until concluded.

```
<body> := <pd>

  <pd> := <<seq>>

    <<seq>> :=  name:"SEQUENTIAL", <pe>

      <pe> := <pd>

        <pd> := <<sim>>

          <<sim>> :=  name:"SIMULTANEOUS", <pe>

            <pe> := name:"VOICE", <voice structure>
                    name:"GRAPHICS", <graphics structure>
```

```
or

proplist:(
  name:"SEQUENTIAL",
    proplist:(
      name:"SIMULTANEOUS",
        proplist:(
          name:"VOICE", <voice structure>
          name:"GRAPHICS", <graphics structure>
        )endlist,
      name:"SIMULTANEOUS",
        proplist:(
          name:"VOICE", <voice structure>
          name:"GRAPHICS", <graphics structure>
        )endlist,
      .
      .
      .
    )endlist
)endlist
```

Such a document structure suggests a slide presentation.

Multiple Active Stream Example

This example is exotic but illustrates what is possible. By making use of the structure and the simultaneous ordering it is possible to start in parallel two or more separate streams. Each stream will proceed at its own pace until all are concluded.

```
<body> := <pd>

  <pd> := name:"SIMULTANEOUS", <pe>

    <pe> = <pd>

      <pd> := name:"SEQUENTIAL", <pe>

        <pe> = <pd>

          <pd> := name:"SIMULTANEOUS", <pe>

            <pe> := name:"VOICE",
                                          <voice structure>
                    name:"GRAPHICS",
                                        <graphics structure>
```

```
or

proplist:(
 name:"SIMULTANEOUS",
   proplist:(
     name:"SEQUENTIAL",
       proplist:(
         name:"SIMULTANEOUS",
           proplist:(
             name:"VOICE", <voice structure>
             name:"GRAPHICS", <graphics structure>
           )endlist,
         name:"SIMULTANEOUS",
            proplist:(
              name:"VOICE", <voice structure>
              name:"GRAPHICS", <graphics structure>
            )endlist,
          .
          .
          .
       )endlist
     name:"SEQUENTIAL",
       proplist:(
         name:"SIMULTANEOUS",
           proplist:(
             name:"VOICE", <voice structure>
             name:"GRAPHICS", <graphics structure>
           )endlist,
          .
          .
          .
       )endlist
   )endlist
)endlist
```

2.3.5. The Media

So far no explicit description has been given for the media classes which fit into a PE. It is not known what types of media will be supported in the various document stations in the future. Those for which support is in part already available are:

TEXT
VOICE
FACSIMILE
GRAPHICS

Standard formats for data in each of these media must be defined.

2.3.6. TEXT

The text data may be structured according to a variety of protocols (yet to be defined). The top level of the data structure is a property list which identifies the protocol, and the version of that protocol.

```
name:"TEXT", proplist:(
                name:"PROTOCOL", <protocol>,
                name:"VERSION", <version>,
                name:"DATA", <data>
              )endlist
```

The first protocol is called PARAGRAPH, and the data is a list of paragraphs, where each paragraph is a text element.

```
name:"DATA", list:(
               text: <paragraph>
               text: <paragraph>
               .
               .
               .
             )endlist
```

2.3.7. VOICE

Since a good deal of research has been done towards implementing the transmission of voice data on the ARPANET, the Network Voice Protocol (NVP) provides the basis for the standard for voice data [24].

Voice data a property list which specifies the vocoder being used, the transmission protocol and the parcel data. The parcel data form is specific to the protocol used and is grouped in lists.

```
name:"VOICE", proplist:(
                 name:"VOCODER", <vocoder>,
                 name:"PROTOCOL", <protocol>,
                 name:"VERSION", <version>,
                 name:"DATA", <data>
               )endlist
```

The NVP protocol has a number of parameters, the version number specifies a certain set of the parameters used by the vocoder hardware and software to set up timing and define the type of coding used. It is not expected that within a document the version number will change.

NVP itself supports negotiation of these parameters to insure both ends of a network speech connection 'understand' one another. Since no such interactive negotiation is possible in a document system, negotiation capabilities have been excluded. As differing hardware becomes available new versions may be defined.

For the NVP protocol the data list will take the following form:

```
name:"DATA", list:(
                    bitstr: <parcel>
                    bitstr: <parcel>
                    .
                    .
                    .
                  )endlist
```

The items in the list are parcels. The individual parcels are bit string data elements whose contents and length are predefined by the version number. The number of parcels in a parcel group is available from the item count in the enclosing list header.

2.3.8. FACSIMILE

There are a number of facsimile devices in use. While standards are being established by CCITT [25], of the devices available today many are incompatible due to proprietary compression algorithms. The description of fax data will allow for the possibility of several protocols.

```
name:"FACSIMILE", proplist:(
                     name:"DEVICE", <device>,
                     name:"PROTOCOL", <protocol>,
                     name:"DATA", <data>
                   )endlist
```

There are few facsimile devices interfaced to computers though, and the existing experiments in the ARPANET all use the RAPICOM 450. A first facsimile standard format will be based on the data structure used for this machine [26]. That is, for device RAPICOM450 and protocol BLOCK, the data will be:

```
name:"DATA", list:(
                    bitstr:<r450-block>,
                    bitstr:<r450-block>,
                    .
                    .
                    .
                  )endlist
```

Where an r450-block is a 585 bit unit.

2.3.9. GRAPHICS

The situation for graphics bears much similarity to facsimile. Devices on the market today have a variety of user interfaces and options. A similar structure is defined.

```
name:"GRAPHICS", proplist:(
                    name:"DEVICE", <device>,
                    name:"PROTOCOL", <protocol>,
                    name:"DATA", <data>
                   )endlist
```

There are several candidate protocols for use in describing graphics data in documents. One is the Network Graphics Protocol [27], another is the Graphics Language [28.29], and a third is the SIGGRAPH Core System [30].

3. EXAMPLES & SCENARIOS

Example 1: Text Example

Suppose we want to send the following message:

```
Date: 1979-03-29-11:46-08:00
From: Jon Postel <Postel@ISIF>
Subject: Meeting Thursday
To: Danny Cohen <Cohen@ISIB>
CC: Linda

Danny:

Please mark your calendar for our meeting Thursday at 3 pm.

--jon.
```

It will be encoded in the structured format. The following will present successive steps in the top down generation of this message. The identification and command portions of the messages will not be expanded here (see [1]).

```
1.  message

2.  (identification, command, document)

3.  (ID:<<identification>>,
     CMD:<<command>>,
     DOC:( date, from, subject, to, cc, body))

4.  (ID:<<identification>>,
     CMD:<<command>>,
     DOC:(DATE:date,
          FROM:from
          SUBJECT:subject,
          TO:to,
          CC:cc,
          BODY:body))

5.  (ID:<<identification>>,
     CMD:<<command>>,
     DOC:(DATE: 1979-03-29-11:46-08:00,
          FROM: (NET:ARPANET,HOST:ISIF,USER:Postel,PERSON:Jon Postel),
          SUBJECT: Meeting Thursday,
          TO: (NET:ARPANET,HOST:ISIB,USER:Cohen,PERSON:Danny Cohen),
          CC: (NET:ARPANET,HOST:ISIF,USER:Linda),
          BODY:
            Danny:
```

```
            Please mark your calendar for our meeting
            Thursday at 3 pm.

            --jon.))
```

6.
```
PROPLIST:
 (ID:<<identification>>,
  CMD:<<command>>,
  DOC:
    PROPLIST:(
      DATE: 1979-03-29-11:46-08:00,
      FROM:
        LIST:(
          PROPLIST:(
            NET:ARPANET,
            HOST:ISIF,
            USER:Postel,
            PERSON:Jon Postel,
           )ENDLIST,
         )ENDLIST,
      SUBJECT: Meeting Thursday,
      TO:
        LIST:(
          PROPLIST:(
            NET:ARPANET,
            HOST:ISIB,
            USER:Cohen,
            PERSON:Danny Cohen,
          )ENDLIST,
        )ENDLIST,
      CC:
        LIST:(
          PROPLIST:(
            NET:ARPANET,
            HOST:ISIF,
            USER:Linda,
          )ENDLIST,
        )ENDLIST,
      BODY:
        Danny:

        Please mark your calendar for our meeting
        Thursday at 3 pm.

        --jon.
    )ENDLIST
)ENDLIST
```

7.
```
proplist:(
  name:"ID", <<identification>>,
  name:"CMD", <<command>>,
  name:"DOC",
    proplist:(
      name:"DATE", name:"1979-03-29-11:46-08:00",
      name:"FROM",
        list:(
          proplist:(
            name:"NET", name:"ARPANET",
            name:"HOST", name:"ISIF",
            name:"USER", name:"Postel",
            name:"PERSON", name:"Jon Postel",
          )endlist,
        )endlist,
      name:"SUBJECT", text:"Meeting Thursday",
      name:"TO",
        list:(
          proplist:(
            name:"NET", name:"ARPANET",
            name:"HOST", name:"ISIB",
            name:"USER", name:"Cohen",
            name:"PERSON", name:"Danny Cohen",
          )endlist,
        )endlist,
      name:"CC",
        list:(
          proplist:(
            name:"NET", name:"ARPANET",
            name:"HOST", name:"ISIF",
            name:"USER", name:"Linda",
          )endlist,
        )endlist,
      name:"BODY",
        text:"Danny:

              Please mark your calendar for our
              meeting Thursday at 3 pm.

              --jon."
    )endlist
)endlist
```

Example 2: Multimedia Example

```
proplist:(
  name:"ID", <<identification>>,
  name:"CMD", <<command>>,
  name:"DOC",
    proplist:(
      name:"DATE", name:"1980-08-06-11:46-08:00",
      name:"FROM",
        list:(
          proplist:(
            name:"NET", name:"ARPANET",
            name:"HOST", name:"ISIF",
            name:"USER", name:"Postel",
            name:"PERSON", name:"Jon Postel",
          )endlist,
        )endlist,
      name:"SUBJECT", text:"Multimedia Test Message",
      name:"TO",
        list:(
          proplist:(
            name:"GROUP", name:"Multimedia Experiment List",
        )endlist,
    )endlist,
  name:"CC",
    list:(
      proplist:(
        name:"NET", name:"ARPANET",
        name:"HOST", name:"ISIF",
        name:"USER", name:"Linda",
      )endlist,
    )endlist,
  name:"BODY",
    proplist:(
      name:"SEQUENTIAL",
        proplist:(
          name:"TEXT",
            proplist:(
              name:"PROTOCOL", name:"PARAGRAPH",
              name:"VERSION", index:"1",
              name:"DATA",
                list:(
                  text:"This is a test of multimedia mail."
                  text:"I hope you like it."
                )endlist
            )endlist
```

```
                name:"SIMULTANEOUS",
                  proplist:(
                    name:"VOICE",
                      proplist:(
                        name:"VOCODER", name:<vocoder>,
                        name:"PROTOCOL", name:"NVP",
                        name:"VERSION", index:"1",
                        name:"DATA",
                          list:(
                            bitstr:<parcel>
                            bitstr:<parcel>
                          )endlist
                      )endlist
                    name:"GRAPHICS",
                      proplist:(
                        name:"DEVICE", name:<device>,
                        name:"PROTOCOL", name:<protocol>,
                        name:"VERSION", index:<version>,
                        name:"DATA",<data>
                          )endlist
                      )endlist
                  )endlist
            name:"SEQUENTIAL",
              proplist:(
                name:"TEXT,
                  proplist:(
                    name:"PROTOCOL", name:"PARAGRAPH",
                    name:"VERSION", index:"1",
                    name:"DATA",
                      list:(
                        text:"That was supposed to be some voice
                              and graphics in parallel."
                        text:"--jon."
                      )endlist
                  )endlist
              )endlist
          )endlist
      )endlist
```

REFERENCES

[1] Postel, J., "Internet Message Protocol," RFC 759, 113, USC/Information Sciences Institute, August 1980.

[2] Bhushan, A., K. Pogran, R. Tomlinson, and J. White, "Standardizing Network Mail Headers," RFC 561, NIC 18516, September 1973.

[3] Myer, T., and D. Henderson, "Message Transmission Protocol," RFC 680, NIC 32116, 30 April 1975.

[4] Crocker, D., J. Vittal, K. Pogran, and D. Henderson, "Standard for the Format of ARPA Network Text Messages," RFC 733, NIC 41952, 21 November 1977.

[5] Barber, D., and J. Laws, "A Basic Mail Scheme for EIN," INWG 192, February 1979.

[6] Braaten, O., "Introduction to a Mail Protocol," Norwegian Computing Center, INWG 180, August 1978.

[7] Crocker, D., E. Szurkowski, and D. Farber, "An Internetwork Memo Distribution Capability - MMDF," Sixth Data Communications Symposium, ACM/IEEE, November 1979.

[8] Haverty, J., D. Henderson, and D. Oestreicher, "Proposed Specification of an Inter-site Message Protocol," 8 July 1975.

[9] Thomas, R., "Providing Mail Services for NSW Users," BBN NSW Working Note 24, Bolt Beranek and Newman, October 1978.

[10] White, J., "A Proposed Mail Protocol," RFC 524, NIC 17140, SRI International, 13 June 1973.

[11] White, J., "Description of a Multi-Host Journal," NIC 23144, SRI International, 30 May 1974.

[12] White, J., "Journal Subscription Service," NIC 23143, SRI International, 28 May 1974.

[13] Levin, R., and M. Schroeder, "Transport of Electronic Messages Through a Network," Teleinformatics 79, Boutmy & Danthine (eds.) North Holland Publishing Co., 1979.

[14] Earnest, L., and J. McCarthy, "DIALNET: A Computer Communications Study," Computer Science Department, Stanford University, August 1978.

[15] Crispin M., "DIALNET: A Telephone Network Data Communications Protocol," DECUS Proceedings, Fall 1979.

[16] Caulkins, D., "The Personal Computer Network (PCNET) Project: A Status Report," Dr. Dobbs Journal of Computer Calisthenics and Orthodontia, v.5, n.6, June 1980.

[17] Postel, J., "NSW Transaction Protocol (NSWTP)," USC/Information Sciences Institute, IEN 38, May 1978.

[18] Haverty, J., "MSDTP -- Message Services Data Transmission Protocol," RFC 713, NIC 34739, April 1976.

[19] ISO-2014, "Writing of calendar dates in all-numeric form," Recommendation 2014, International Organization for Standardization, 1975.

[20] ISO-3307, "Information Interchange -- Representations of time of the day," Recommendation 3307, International Organization for Standardization, 1975.

[21] ISO-4031, "Information Interchange -- Representation of local time differentials," Recommendation 4031, International Organization for Standardization, 1978.

[22] Postel, J., "DOD Standard Internet Protocol," USC/Information Sciences Institute, IEN 128, NTIS number AD A079730, January 1980.

[23] CCITT-X.121, "International Numbering Plan for Public Data Networks," Recommendation X.121, CCITT, Geneva, 1978.

[24] Cohen, D., "Specifications for the Network Voice Protocol (NVP)," NIC 42444, RFC 741, NSC 68, RR-75-39, USC/Information Sciences Institute, January 1976.

[25] CCITT-T.30, "Procedures for Document Facsimile Transmission in the General Switched Telephone Network," Recommendation T.30, Orange Book, V. 7, The International Telephone and Telegraph Consulative Committee, International Telecommunication Union, Geneva, 1977.

[26] Treadwell, S., "FAX File Format," ARPANET Message, 14 November 1979.

[27] Sproull, R., and E. Thomas, "A Network Graphics Protocol," NIC 24308, Xerox Palo Alto Research Center, August 1974.

[28] Bisbey, R., and D. Hollingworth, "A Distributable, Display-Device-Independent Vector Graphics System for Command and Control," RR-80-87, USC/Information Sciences Institute, July 1980.

[29] Bisbey, R., D. Hollingworth, and B. Britt, "Graphics Language," TM-80-18, USC/Information Sciences Institute, July 1980.

[30] Graphics Standard Planning Committee, "Core System," Computer Graphics, V. 13, N. 3, SIGGRAPH, ACM, August 1979.